МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ   
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ**

**ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»**

**(БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

**Лабораторная работа №2**

по дисциплине: Вычислительная математика

тема: «*Алгебра матриц. Быстрое умножение матриц. Вычисление обратной матрицы. Нахождение собственных чисел и собственных векторов матрицы*»

Выполнил: ст. группы ПВ-233

Ситников Алексей Павлович

Проверил:

Горбов Даниил Игоревич

Белгород 2025 г.

**Цель работы**: изучить алгебраические операции над матрицами, особенности алгоритмизации быстрых матричных алгоритмов (на примере умножения матриц), вычисления обратной матрицы, нахождения собственных чисел и собственных векторов матрицы.

Эмпирически оценить временную сложность функции dot для умножения матриц из библиотеки NumPy (Python).

Код:

import numpy as np  
import time  
import matplotlib.pyplot as plt  
  
  
def measure\_dot\_time(size, repetitions=5):  
 A = np.random.rand(size, size)  
 B = np.random.rand(size, size)  
  
 total\_time = 0  
 for \_ in range(repetitions):  
 start\_time = time.perf\_counter()  
 np.dot(A, B)  
 end\_time = time.perf\_counter()  
 total\_time += (end\_time - start\_time)  
  
 return total\_time / repetitions  
  
  
sizes = [1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000]  
times = []  
  
for size in sizes:  
 elapsed\_time = measure\_dot\_time(size)  
 times.append(elapsed\_time)  
 print(f"Size: {size}, Time: {elapsed\_time:.4f} seconds")  
  
# Построение графика  
plt.plot(sizes, times, marker='o')  
plt.xlabel('Размер матрицы (n x n)')  
plt.ylabel('Время выполнения (секунды)')  
plt.title('Время выполнения np.dot для умножения матриц')  
plt.grid()  
plt.show()

Итог:  
Изображение выглядит как текст, Шрифт, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, линия, График

Контент, сгенерированный ИИ, может содержать ошибки.

По графику видно, что сложность между O(n2) и O(n3).

Вариант 13

Нахождение обратной матрицы вручную:  
Изображение выглядит как текст, рукописный текст, доска, Самоклеющийся листок

Контент, сгенерированный ИИ, может содержать ошибки.

Нахождение обратной матрицы методом *Ньютона-Шульца,* и с использованием np.linalg.inv.

Код:

import numpy as np  
  
  
def compute\_inverse\_newton\_schulz(matrix, num\_iterations=5):  
  
  
 assert matrix.shape[0] == matrix.shape[1], "Матрица не квадратная"  
  
  
 initial\_approximation = matrix.T / (np.linalg.norm(matrix) \*\* 2)  
 # Единичная матрица того же размера, что и A  
 identity\_matrix = np.eye(matrix.shape[0])  
  
  
 for \_ in range(num\_iterations):  
 initial\_approximation = initial\_approximation @ (2 \* identity\_matrix - matrix @ initial\_approximation)  
  
 return initial\_approximation  
  
  
  
matrix = np.array([[56, -32, 14],  
 [-23, 59, -10],  
 [40, -67, 21]], dtype=np.float64)  
  
print("Исходная матрица:")  
print(matrix)  
  
  
inverse\_matrix\_ns = compute\_inverse\_newton\_schulz(matrix, num\_iterations=10)  
print("\nОбратная матрица, полученная методом Ньютона-Шульца:")  
print(inverse\_matrix\_ns)  
  
  
inverse\_matrix\_np = np.linalg.inv(matrix)  
print("\nОбратная матрица, полученная с помощью np.linalg.inv:")  
print(inverse\_matrix\_np)

Вывод:

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

Решение совпало с вычисленным вручную.

Нахождение собственных чисел и векторов:

import numpy as np  
  
A = np.array([[56, -32, 14],  
 [-23, 59, -10],  
 [40, -67, 21]], dtype=np.float64)  
  
  
print("Исходная матрица:\n", A)  
eigenvalues, eigenvectors = np.linalg.eig(A)  
print("Собственные числа матрицы:\n", eigenvalues)  
print("Собственные вектора матрицы:\n", eigenvectors)

Вывод:  
Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

**Вывод:** я изучил алгебраические операции над матрицами, особенности алгоритмизации быстрых матричных алгоритмов (на примере умножения матриц), вычисления обратной матрицы, нахождения собственных чисел и собственных векторов матрицы.